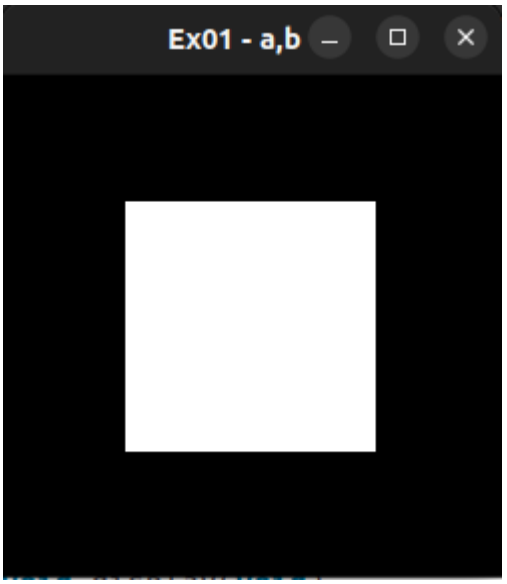
Universidade Federal do ABC
Fundamentos de Processamento Gráfico 2023.3
Prof. Celso Setsuo Kurashima
Andressa Guimaraes Benedicto 11201810280
Heitor Rodrigues Savegnago 11077415
Kaleb Lucas da Silva Alves 21049916
Pedro Domingos Napole Certo 11201722682

Laboratório 02 - Prática com OpenGL

Através da API gráfica OpenGL e sua estrutura mínima de programa, é possível renderizar um quadrado simples bidimensional utilizando os comandos: glColor(), glVertex(), glBegin(), glEnd() e glFlush(), passando como argumento as tuplas de vértices e cores, bem como o objeto a ser renderizado ao comando glBegin():

```
glClear (GL_COLOR_BUFFER_BIT);
glColor3f (1.0, 1.0, 1.0);
glBegin(GL_POLYGON);
    glVertex3f (0.25, 0.25, 0.0);
    glVertex3f (0.75, 0.25, 0.0);
    glVertex3f (0.75, 0.75, 0.0);
    glVertex3f (0.25, 0.75, 0.0);
glEnd();
glFlush ();
```

Resultado (itens *a* e *b*):

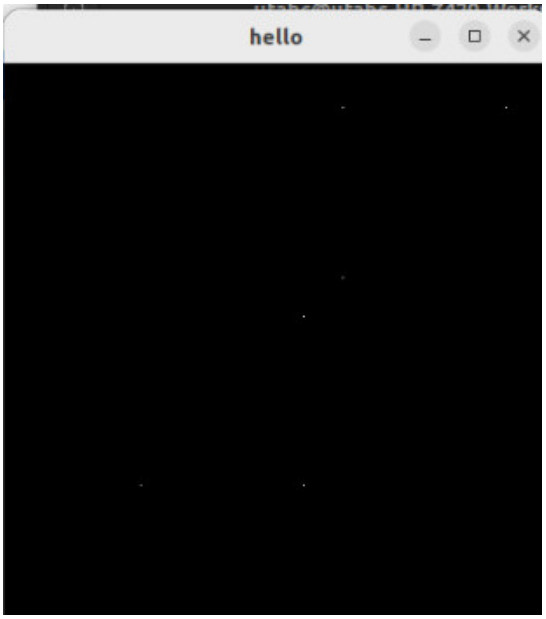


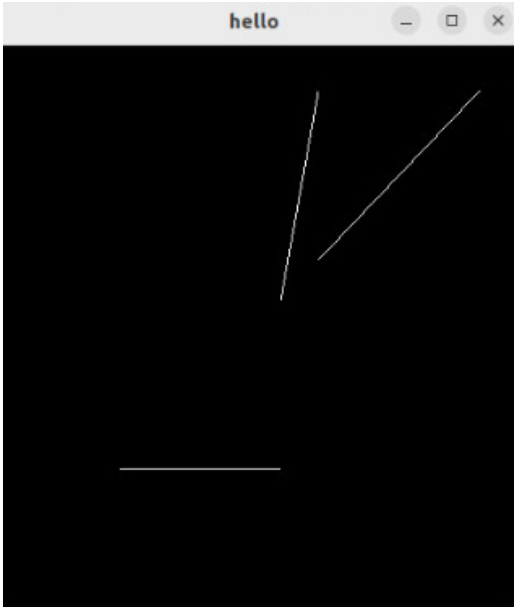
Passando diferentes constantes da API para a função glBegin(), é possível gerar diferentes imagens com os mesmos vértices e cores definidos:

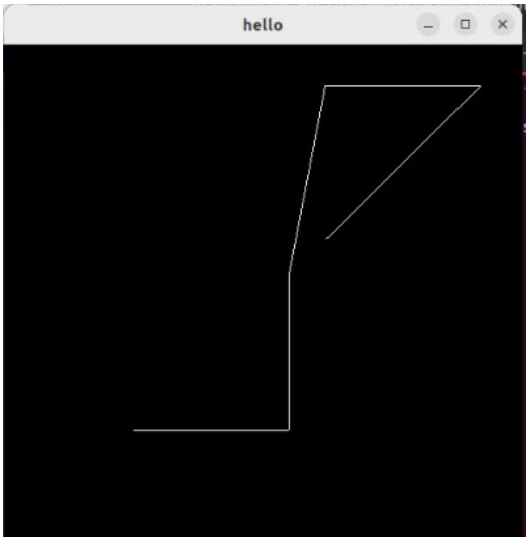
EXERCÍCIO C

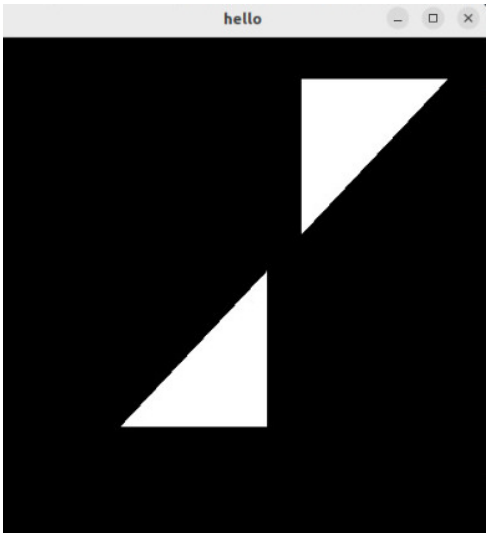
Para que ficasse mais claro à diferença entre cada um dos comandos, alteramos o código de maneira a formar dois triângulos, e à partir dessas coordenadas fomos aplicando as primitivas

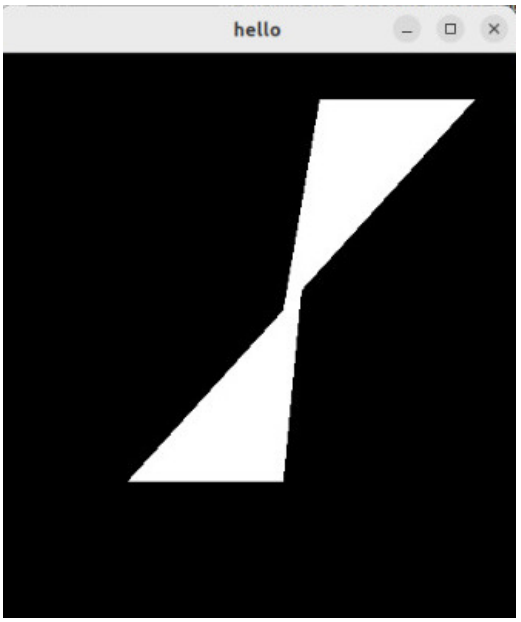
```
glColor3f (1.0, 1.0, 1.0);
glBegin(GL_LINE_STRIP);
    glVertex3f (0.25, 0.25, 0.0);
    glVertex3f (0.55, 0.25, 0.0);
    glVertex3f (0.55, 0.55, 0.0);
    glVertex3f (0.62, 0.92, 0.0);
    glVertex3f (0.92, 0.92, 0.0);
    glVertex3f (0.62, 0.62, 0.0);
glEnd();
```

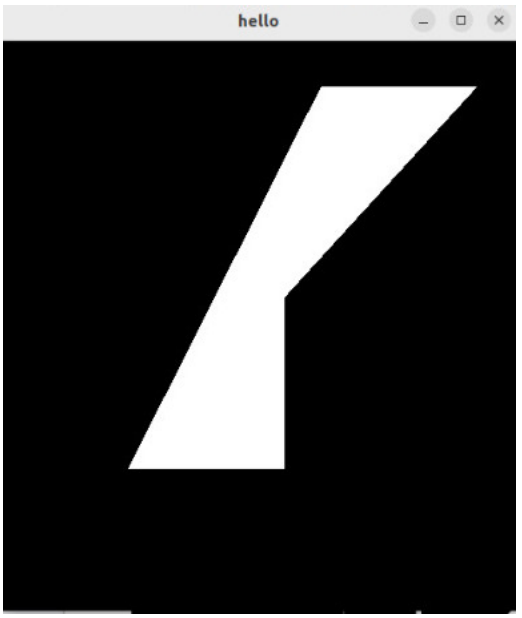


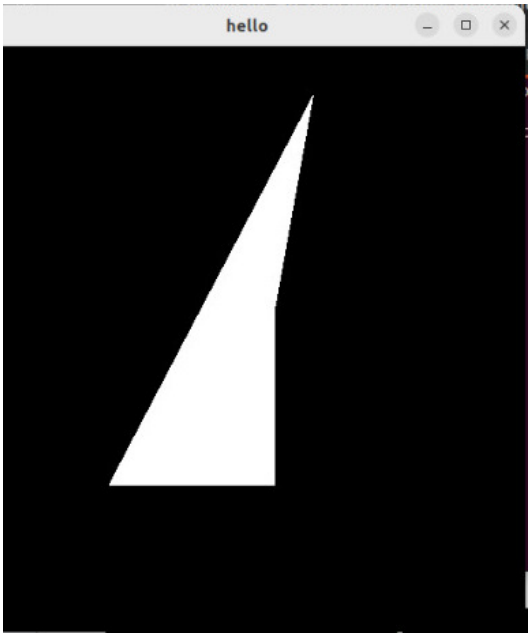




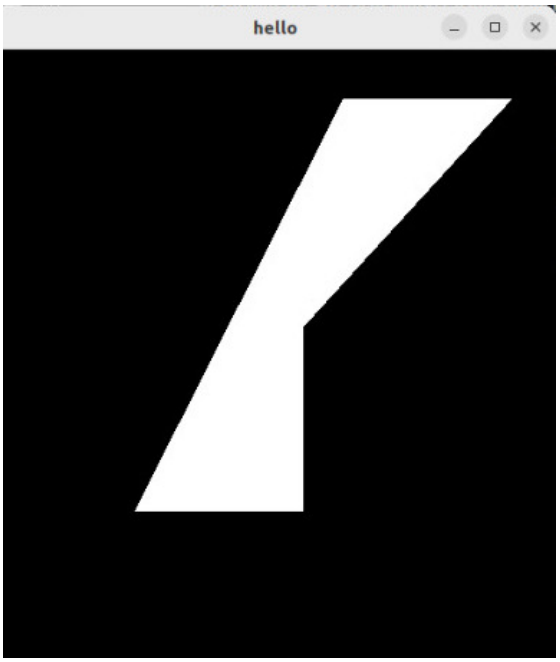



hello


hello

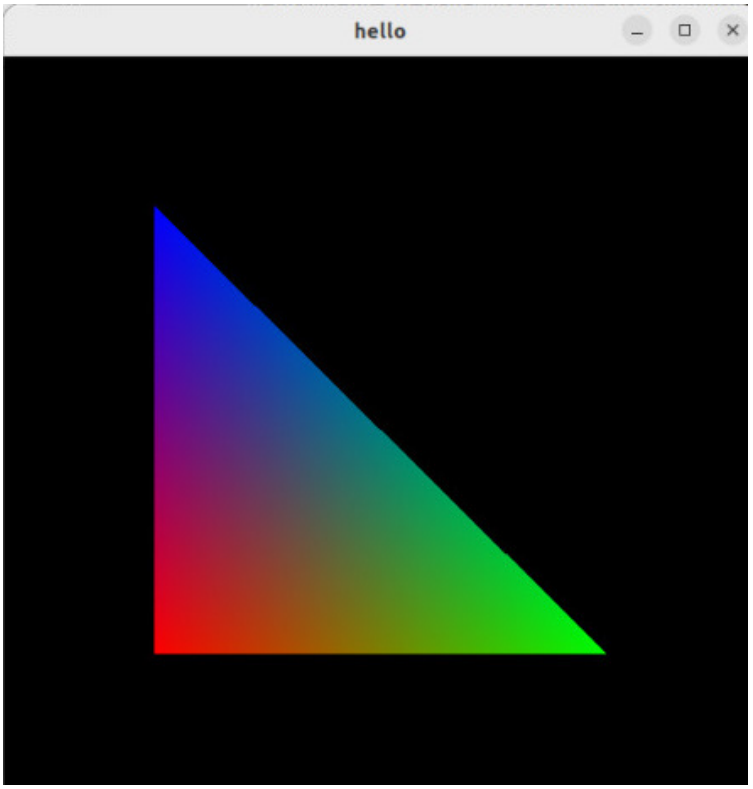

hello


hello

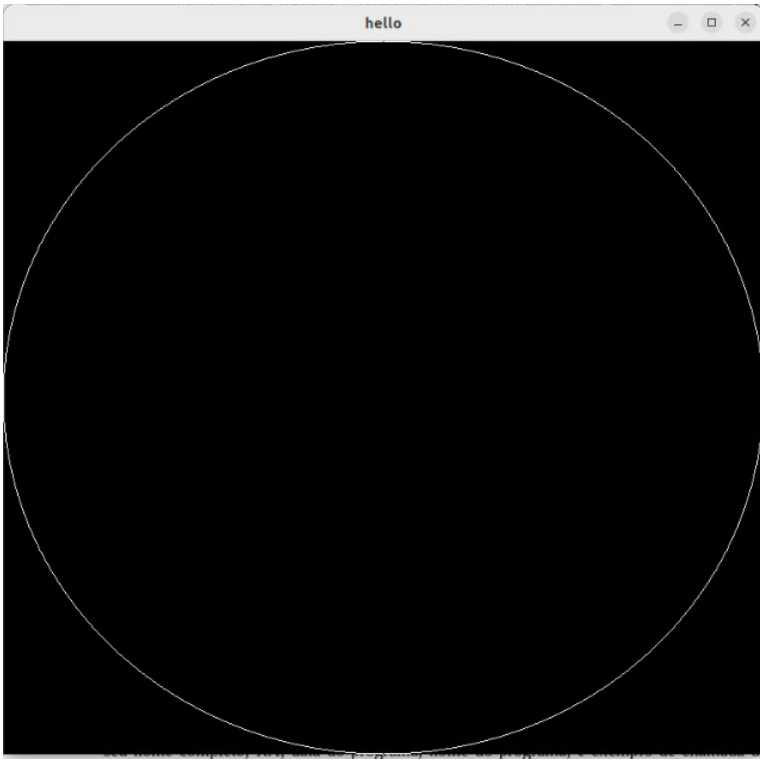

hello

Os resultados acima foram obtidos utilizando as primitivas, GL_POINTS, GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP, GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_QUADS, GL_QUAD_STRIP, GL_POLYGON, respectivamente.

EXERCÍCIO D



EXERCÍCIO E

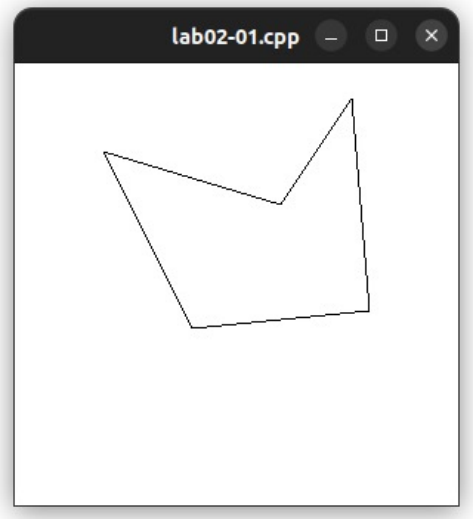


EXERCÍCIOS PARA ENTREGAR

1-

```
/*
 * lab02-01.cpp
 * Heitor Rodrigues Savegnago - 11077415 - 2023-10-10
 * gcc lab02-02.cpp -o /tmp/temp.run -lm -lglut -lGL -lGLU && /tmp/temp.run
 */
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
#include <stdlib.h>
#include <math.h>

// Função callback chamada para fazer o desenho
void Desenha(void)
{
	// Limpa a janela de visualização com a cor de fundo especificada
	glClear(GL_COLOR_BUFFER_BIT);

	glBegin(GL_LINE_LOOP);

	glColor3f(0.0, 0.0, 0.0);
	glVertex2f(0.0f, 0.0f);
	glVertex2f(0.25f, -0.5f);
	glVertex2f(0.75f, -0.45f);
	glVertex2f(0.7f, 0.15f);
	glVertex2f(0.5f, -0.15f);
	glEnd();

	// Executa os comandos OpenGL
	glFlush();
}
// Inicializa parâmetros de rendering
void Inicializa(void)
{
	// Define a cor de fundo da janela de visualização como preta
	glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
	glOrtho(-0.25, 1, -1, 0.25, -1.0, 1.0);
}
// Programa Principal
int main(int argc, char *argv[])
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutCreateWindow("lab02-01.cpp");
	glutDisplayFunc(Desenha);
	Inicializa();
	glutMainLoop();
	return 0;
```

}

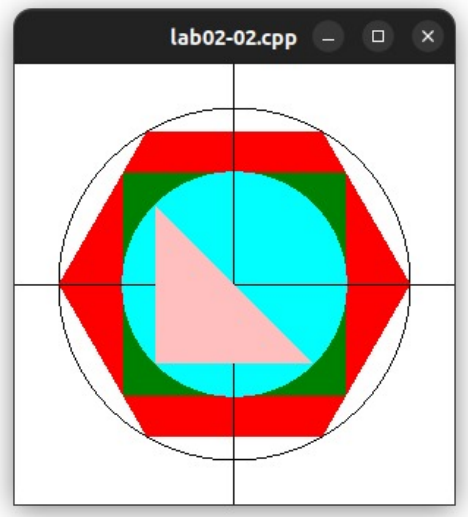


2-

```
/*
 * lab02-02.cpp
 * Heitor Rodrigues Savegnago - 11077415 - 2023-10-10
 * gcc lab02-02.cpp -o /tmp/temp.run -lm -lglut -lGL -lGLU && /tmp/temp.run
 */
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
#include <stdlib.h>
#include <math.h>

// Função callback chamada para fazer o desenho
void Desenha(void)
{
	// Limpa a janela de visualização com a cor de fundo especificada
	glClear(GL_COLOR_BUFFER_BIT);

	// glBegin(GL_POLYGON);
	//	glVertex3f(0.25f, 0.25f, 0.0f);
	//	glVertex3f(0.75f, 0.25f, 0.0f);
	//	glVertex3f(0.75f, 0.75f, 0.0f);
	//	glVertex3f(0.25f, 0.75f, 0.0f);
	// glEnd();

	// glBegin(GL_TRIANGLES);
	//	glColor3f(1.f, 0.f, 0.f);
	//	glVertex3f(0.2f, 0.2f, 0.f);
	//	glColor3f(0.f, 1.f, 0.f);
	//	glVertex3f(0.8f, 0.2f, 0.f);
	//	glColor3f(0.f, 0.f, 1.f);
	//	glVertex3f(0.2f, 0.8f, 0.f);
	// glEnd();
```

```
GLdouble PI = 3.1415926535897;
GLint circle_points = 100;
GLdouble radius = 1;

glBegin(GL_LINE_LOOP); // Circulo
for (int i = 0; i < circle_points; i++)
{
float angle = 2 * PI * i / circle_points;
glColor3f(0.0, 0.0, 0.0);
glVertex2f(radius*cos(angle), radius*sin(angle));
}
glEnd();

glBegin(GL_POLYGON); //Hexagono
for (int i = 0; i < 360; i += 60 )
{
float angle = i * PI / 180.0;
glColor3f(1.0, 0.0, 0.0);
glVertex2f(radius*cos(angle), radius*sin(angle));
}
glEnd();

radius = 0.9;

glBegin(GL_POLYGON); //Quadrado
for (int i = 45; i < 360; i += 90 )
{
float angle = i * PI / 180.0;
glColor3f(0.0, 0.5, 0.0);
glVertex2f(radius*cos(angle), radius*sin(angle));
}
glEnd();

radius = 0.64;

glBegin(GL_POLYGON); // Circulo
for (int i = 0; i < circle_points; i++)
{
float angle = 2 * PI * i / circle_points;
glColor3f(0.0, 1.0, 1.0);
glVertex2f(radius*cos(angle), radius*sin(angle));
}
glEnd();

glBegin(GL_LINES); // Circulo
// for (int i = 0; i < circle_points; i++)
// {
// float angle = 2 * PI * i / circle_points;
glColor3f(0.0, 0.0, 0.0);
glVertex2f(-1.25,0);
glVertex2f(1.25,0);
glVertex2f(0,-1.25);
glVertex2f(0,1.25);
```

```
        // }
        glEnd();

        glBegin(GL_POLYGON); //Triangulo retangulo
        for (int i = 135; i < 360; i += 90 )
        {
        float angle = i * PI / 180.0;
        glColor3f(1.0, 0.75, 0.75);
        glVertex2f(radius*cos(angle), radius*sin(angle));
        }
        glEnd();

        // Executa os comandos OpenGL
        glFlush();
}
// Inicializa parâmetros de rendering
void Inicializa(void)
{
        // Define a cor de fundo da janela de visualização como preta
        // glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
        glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
        glOrtho(-1.25, 1.25, -1.25, 1.25, -1.25, 1.25);
}
// Programa Principal
int main(int argc, char *argv[])
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutCreateWindow("lab02-02.cpp");
        glutDisplayFunc(Desenha);
        Inicializa();
        glutMainLoop();
        return 0;
}
```



Referências: